Attorney's Doc[redacted]o.: 06618/503001 / CIT 3057

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: INTERLEAVED SERIAL CONCATENATION FORMING TURBO-LIKE CODES

APPLICANT: DARIUSH DIVSALAR, ROBERT J. MCELIECE, AND HUI JIN

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL558600041US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

August 18, 2000
Date of Deposit

Signature

Derek W. Norwood
Typed or Printed Name of Person Signing Certificate

INTERLEAVED SERIAL CONCATENATION FORMING TURBO-LIKE CODES

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims benefit of U.S. Provisional Application No. 60/149,871, filed August 18, 1999.

The work described herein may have been supported by Grant Nos. NCR 9505975, awarded by the National Science Foundation, and 5F49620-97-1-0313 awarded by the Air Force. The US Government may have certain rights to this invention.

BACKGROUND

Properties of a channel affect the amount of data that can be handled by the channel. The so-called "Shannon limit" defines the theoretical limit of amount of data that a channel can carry.

Different techniques have been used to increase the data rate that can be handled by a channel. "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," by Berrou *et al.* ICC, pp 1064-1070, (1993), described a new "turbo code" technique that has revolutionized the field of error correcting codes.

Turbo codes have sufficient randomness to allow reliable communication over the channel at a high data rate near

capacity. However, they still retain sufficient structure to allow practical encoding and decoding algorithms. Still, the technique for encoding and decoding turbo codes can be relatively complex.

A standard turbo coder is shown in Figure 1. A block of k information bits 100 is input directly to a first encoder 102. A k bit interleaver 110 also receives the k bits and interleaves them prior to applying them to a second encoder 104. The second encoder produces an output that has more bits than its input, that is, it is a coder with rate that is less than 1.

The encoders 102, 104 are also typically recursive convolutional coders.

Three different items are sent over the channel 150: the original k bits 100, first encoded bits 110, and second encoded bits 112.

At the decoding end, two decoders are used: a first constituent decoder 160 and a second constituent decoder 162. Each receives both the original k bits, and one of the encoded portions 110, 112. Each decoder sends likelihood estimates of the decoded bits to the other decoders. The estimates are used to decode the uncoded information bits as corrupted by the noisy channel.

SUMMARY

The present application describes a new class of codes, coders and decoders: called "turbo-like" codes, coders and decoders. These coders may be less complex to implement than standard turbo coders.

The inner coder of this system is rate 1 encoder, or a coder that encodes at close to rate 1. This means that this coder puts out a similar number of bits to the number it takes in. Fewer bits are produced as compared with other systems that use rate less than 1 as their inner coder.

The system can also use codes use component codes in a serially concatenated system. The individual component codes forming the overall code may be simpler than previous codes. Each simple code individually might be considered useless.

More specifically, the present system uses an outer coder, an interleaver, and inner coder. Optional components include a middle coder 305, where the middle coder can also include additional interleavers.

The inner coder 200 is a linear rate 1 coder, or a coder whose rate is close to 1.

Unlike turbo coders that produce excess information in their final coder, the present system uses a final coder which does not increase the number of bits. More specifically,

however, the inner coder can be one of many different kinds of elements.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a prior "turbo code" system;

Figure 2 shows a generic turbo-like coder in its most general form with a single rate 1 inner coder, single outer coder, and a single interleaver;

Figure 3 shows a x = 4 coder;

Figures 4 and 5 show a repeat and accumulate coder;

Figure 6 shows a repeat/double accumulator coder;

Figure 7 shows a dual accumulator system;

Figure 8 shows a tree structure with a second branch;

Figure 9 shows a flow chart of Tanner Graph decoding; and

Figure 10 shows the actual Tanner Graph decoding.

DETAILED DESCRIPTION

An embodiment of the present system, in its most general form, is shown in Figure 2. In general, this system has two encoders: an outer coder 200 and an inner coder 210 separated by an interleaver 220.

Encoder 200 is called an outer encoder, and receives the uncoded data. The outer coder can be an (n,k) binary linear

encoder where n > k. The means that the encoder 200 accepts as input a block u of k data bits. It produces an output block v of n data bits. The mathematical relationship between u and v is $v=T_0u$, where T_0 is an n x k binary matrix. In its simplest form, the outer coder may be a repetition coder. The outer coder codes data with a rate that is less than 1, and may be, for example, 1/2 or 1/3.

Sub a4 The interleaver 220 performs a fixed pseudo-random permutation of the block v, yielding a block w having the same length as v. The permutation can be an identity matrix, where the output becomes identically the same as the input. Alternately and more preferably, the permutation rearranges the bits in a specified way.

The inner encoder 210 is a linear rate 1 encoder, which means that the n-bit output block x can be written as $x=T_Iw$, where T_I is a nonsingular n x n matrix. Encoder 210 can have a rate that is close to 1, *e.g.*, within 50%, more preferably 10% and perhaps even more preferably within 1% of 1.

The overall structure of coders such as the one in Figure 8 has no loops, *i.e.*, it is not "recursive" between coders. The whole operation proceeds by a graph theoretic tree. A tree structure can simplify the overall operation.

A number of different embodiments will be described herein, all of which follow the general structure of Figure 2 which includes the first outer coder 200 (rate <1), which can be an encoder for a binary (n,k) linear block code; a pseudo random interleaver 220 which receives the output (rate 1), and a rate 1 inner coder 210 that codes the interleaved output.

More generally, there can be more than 2 encoders: there can be x encoders, and x-1 interleavers. The additional coder can be generically shown as a middle coder. Figure 3 shows four encoders 300, 310, 320, 330. Three of these coders; here 310, 320, 330; are rate 1 encoders. The outer encoder 300 is an (n,k) linear block coding encoder. Three pseudorandom interleavers 340, 350, 360 separate the rate 1 coders from the outer coder 300. The middle coder, in general, has a rate less than or equal to 1.

A number of embodiments of the coders are described including a repeat and accumulate ("RA") coder, an repeat double accumulate ("RDD") coder and a repeat accumulate accumulate ("RAA") coder.

The RA coder includes an outer coder and an inner coder connected via a pseudorandom interleaver. The outer code uses a simple repetition code, and the inner code is a rate 1 accumulator code. The accumulator code is a truncated rate 1

convolutional code with transfer function 1/(1+D). Further details are provided in the following.

Figures 4 and 5 show two versions of encoder systems for the basic repeat and accumulate code, using the general structure described above. An information block 400 of length k is input to the outer coder 405, here a rate 1/q repetition element. The device 405 replicates the input block q times to produce an information block 410 of length qk. The replication may be carried out a subblock at a time. information 410 is then interleaved by a qk x qk permutation matrix to form information block of length qk 420. This block is then encoded by an accumulator 425. In Figure 5, this accumulator 510 is a truncated rate 1 recursive convolutional coder with transfer function 1/(1+D). Looking at this accumulator mathematically, it can be seen as a block code whose input block $\{x_1,\ldots,x_n\}$ and output block $\{y_1,\ldots,y_n\}$ are related by the formula

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_1 \oplus x_2 \\ y_3 &= x_1 \oplus x_2 \oplus x_3 \\ &\vdots \\ y_n &= x_1 \oplus x_2 \oplus x_3 + \cdots + x_n \end{aligned}$$

In the q = 3 embodiment of the encoder, a block of k data bits $(u[1],u[2],\ldots,u[k])$, (the u-block) is subjected to a

three-stage process which produces a block of $3k$ encoded bits $(x[1],x[2],\ldots,x[3k])$ (the x-block). This process is depicted in Figure 5.

Stage 1 of the encoding process forms the outer encoder stage. This system uses a repetition code. The input "u" block $(u[1],\ldots,u[k])$ is transformed into a $3k$-bit data block $(v[1],v[2],\ldots,v[3k])$ (the v-block). This is done by repeating each data bit 3 times, according to the following rule:

$$\begin{aligned} v[1] = v[2] = v[3] &= u[1] \\ v[4] = v[5] = v[6] &= u[2] \\ \vdots \qquad\qquad & \vdots \\ v[3k-2] = v[3k-1] &= u[3k] = u[k]. \end{aligned}$$

Stage 2 of the encoding process is the interleaver 510. The interleaver converts the v-block into the w-block as follows:

$$\begin{aligned} w[1] &= v[\pi[1]] \\ w[2] &= v[\pi[2]] \\ &\vdots \\ w[3k] &= v[\pi[3k]], \end{aligned}$$

and $\pi[1],\pi[2],\ldots,\pi[3k]$ is a fixed permutation of the set $\{1,2,\ldots,kq\}$ for this case of q=3.

Stage 3 of the encoding process is the accumulator 520. This converts the w-block into the x-block by the following rule:

$$x[1] = w[1]$$
$$x[2] = x[1] \oplus w[2]$$
$$x[3] = x[2] \oplus w[3]$$
$$\vdots$$
$$x[kq] = x[kq - 1] \oplus w[kq],$$

Where "⊕" denotes modulo two, or exclusive or, addition. An advantage of this system is that only mod 2 addition is necessary for the accumulator. That means that the accumulator can be embodied using only exclusive or (xor) gates. This can simplify the design.

The accumulator 520 can alternatively be represented as a digital filter with transfer function equal to 1/(1 + D) as shown in 425.

The RA coder is a 1/q coder, and hence can only provide certain rates, *e.g.* ½, 1/3, 1/4, 1/5, etc. Other variations of this general system form alternative embodiments that can improve performance and provide flexibility in the desired rate.

One such is the "RDD" code. The encoder for RDD is shown in Figure 6. The accumulator component of the RA code is replaced by a "double accumulator." The double accumulator can be viewed as a truncated rate 1 convolutional code with transfer function $1/(1 + D + D^2)$.

In another preferred embodiment shown in Figure 7, called the "RAA" code, there are three component codes: The "outer" code, the "middle" code, and the "inner" code. The outer code is a repetition code, and the middle and inner codes are both accumulators. The outer code has rate less than 1, the middle code are both accumulators (of rate 1) and the inner code has a rate which is 1 or close to 1.

As described above, the "repetition number" q of the first stage of the encoder can be any positive integer greater than or equal to 2. The outer encoder is the encoder for the $(q, 1)$ repetition code.

The outer encoder can carry out coding using coding schemes other than simple repetition. In the most general embodiment, the outer encoder is a (q, k) block code. For example, if k is a multiple of 4, the input block can be partitioned into four bit subblocks, and each 4-bit subblock can be encoded into 8 bits using an encoder for the (8,4) extended Hamming code. Any other short block code can be used in a similar fashion, for example a (23, 12) Golay code.

In general, k can be partitioned into subblocks k_1, k_2, … k_m such that $\sum_{i=1}^{m} k_i = k$. q can be similarly partitioned. This, the k input bits can be encoded by m block codes (q_i, k_i) for any i. In general, these outer codes can be different.

Truncated convolutional codes can be used as the block codes. Repetition codes can also be used as the block codes.

In a similar fashion, the q output bits of the interleaver can be partitioned into j subblocks q'_1, q'_2... such that the summation of all the q'_I =q. Then each subblock can be encoded with a rate 1 inner code. In general these inner codes can be different recursive rate 1 convolutional codes.

The accumulator 520 in stage 3 of the encoder can be replaced by a more general device, for example, an arbitrary digital filter using modulo 2 arithmetic with infinite impulse response ("i.i.r."). Figure 6 shows, for example, the accumulator being an i.i.r. filter with whose transfer function is $1/(1 + D + D^2)$.

The system can be a straight tree, or a tree with multiple branches. Figure 8 shows a multiple branch tree, where the outer encoder c1 feeds two interleavers p3, p4, each of which is associated with a rate 1 inner coder c3, c4. A totally separate branch has the interleaver p2, and rate 1 inner coder c2.

Some or all of the output bits from the outer encoder can be sent directly to the channel and/or to a modulator for the channel.

Any of a number of different techniques can be used for decoding such a code. For example, soft input soft output can be used with a posteriori probability calculations to decode the code.

A specific described decoding scheme relies on exploiting the Tanner Graph representation of an RA code.

Figure 9 shows a flowchart of operation. The code is received, and a Tanner Graph is used to describe the essential structure of the code on a graph at 800.

Roughly speaking, a Tanner Graph G = (V,E) is a bipartite graph whose vertices can be partitioned into variable nodes V_m and check nodes V_c, where edges $E \subseteq V_m \times V_c$. Check nodes in the Tanner Graph represent certain "local constraints" on a subset of variable nodes. An edge indicates that a particular variable is present in a particular constraint.

The Tanner Graph realization for an RA code is explained with reference to Figure 10. For a repetition q type RA code with block length k, the k information bits can be denoted by i= 1,2, ...n, the qk code bits by y_i, and the intermediate bits (which are the outputs of the outer code and the inputs to the inner code) by x_i. y_i and x_i are related by the formula

$$y_i = \begin{cases} x_i & \text{if } i = 1, \\ x_i + y_{i-1} & \text{otherwise.} \end{cases}$$

Notice that every x_i is a replica of some u_j. Therefore, all qk equations in the above can be represented by check nodes c_i. These check nodes represent both information bits u_i and code bits y_i by variable nodes with the same symbol.

Edges can be naturally generated by connecting each check node to the u_i and y_is that are present in its equation. Using notation C ={c_i}, U ={u_i} Y = {y_i} provides a Tanner Graph representation of an RA code, with $V_m = U \bigcup Y$ and V_c = C.

Figure 10 shows such a Tanner Graph specifically for a q=3, k=2 (repetition 3 block length 2) RA code, with permutation π = (1, 2, 5, 3, 4, 6). This graph also shows the received version of code bits y through the channel, which are denoted by y_r. Although the received bits y_r may provide evidence or confirmation in the decoding procedure, they are not strictly part of the Tanner Graph.

Generally, in the Tanner Graph for a repetition q RA code, every u_i is present in q check nodes regardless of the block length k. Hence every vertex $u \in$ U has degree q. Similarly, every vertex $c \in$ C has degree 3 (except the first vertex c_1 which has degree 2), and every vertex $y \in$ Y has degree 2 (except the last vertex y_{qk}, which has degree 1.

"Belief propagation" on the Tanner Graph realization is used to decode RA codes at 910. Roughly speaking, the belief propagation decoding technique allows the messages passed on an edge *e* to represent posterior densities on the bit associated with the variable node. A probability density on a bit is a pair of non-negative real numbers p_o, p_1 satisfying $p_o + p_1 = 1$, where p_o denotes the probability of the bit being 0, p_1 the probability of it being 1. Such a pair can be represented by its log likelihood ratio $\log \frac{p_1}{p_o}$. It can be assumed that the messages here use this representation.

There are four distinct classes of messages in the belief propagation decoding of RA codes, namely messages sent (received) by some vertex $u \in U$ to (from) some vertex $c \in C$, which are denoted by *m[u,c] (m[c,u])*, and messages sent (received) by some vertex $y \in Y$ to (from some vertex $c \in C$, which are denoted by *m[y,c] (m[c,y])*. Messages are passed along the edges, as shown in FIG. 10. Both *m[u,c]* and *m[c,u]* have the conditional value of $\log \frac{p(u=1)}{p(u=0)}$, both *m[y,c]* and *m[c,y]* have the conditional value of $\log \frac{p(y=1)}{p(y=0)}$. Each code node of *y* also has the belief provided by received bit y_r,

which value is denoted by $B(y) = \log \frac{p(y=1/y_r)}{p(y=0/y_r)}$. With all the notations introduced, the belief propagation decoding of an RA code can be described as follows:

Initialize all messages m[u,c], m[c,u], m[y,c],m[c,y] to be zero at 905. Then interate at 910. The messages are continually updated over K rounds at 920 (the number K is predetermined or is determined dynamically by some halting rule during execution of the algorithm). Each round is a sequential execution of the following script:

– Update **m[y,c]**:

$$m[y,c] = \begin{cases} B(y) & \text{if } y = y_{qk}, \\ B(y) + m[c',y] & \text{otherwise, where } (c',y) \in \text{E and c}' \neq c. \end{cases}$$

– Update **m[c,u]**:

$$m[c,u] = \begin{cases} m[y,c] & \text{if } c = c_1, \text{ where } (y,c) \in E \text{ and } y \in Y, \\ \log \frac{e^{m[y,c]} + e^{m[y',c]}}{1 + e^{m[y,c]+m[y',c]}} & \text{otherwise, where } (y,c),(y',c) \in E \text{ and } y \neq y' \in Y. \end{cases}$$

– Update **m[u,c]**:

$$m[u,c] = \sum_{c'} m[u,c'], \text{ where } (u,c') \in E \text{ and c}' \neq c.$$

– Update **m[c,y]**:

$$m[c,y]=\begin{cases} m[u,c] & \text{if } c=c_1, \text{ where } (u,c)\in E \text{ and } u\in U, \\ \log\dfrac{e^{m[u,c]}+e^{m[y',c]}}{1+e^{m[u,c]+m[y',c]}} & \text{otherwise, where } (u,c),(y',c)\in E \text{ and } y\neq y'\in Y. \end{cases}$$

Upon completion of the K iterative propagations, the values are calculated based on votes at 930. Specifically, compute $S_u=\sum_c m[u,c]$ for every $u \in U$, where the summation is over all the c such that (u,c) ∈ E. If s(u) >= 0, bit u is decoded to be 1; otherwise, it is decoded to be 0.

Although only a few embodiments have been disclosed herein, other modifications are possible. For example, the inner coder is described as being close to rate 1. If the rate of the inner coder is greater than one, certain bits can be punctured to decrease the bit rate.